

Docket No.: 62807-150



PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	:	Customer Number: 20277
Atsushi Otake	:	Confirmation Number: 2752
Serial No.: 10/715,121	:	Group Art Unit: 2143
Filed: November 18, 2003	:	Examiner: Unknown
For: PROGRAM CHANGING METHOD	:	

TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT

Mail Stop CPD
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

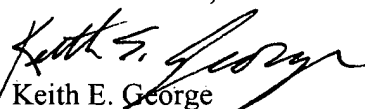
At the time the above application was filed, priority was claimed based on the following application:

Japanese Patent Application No. 2003-057936, filed March 5, 2003

A copy of the priority application listed above is enclosed.

Respectfully submitted,

MCDERMOTT, WILL & EMERY


Keith E. George
Registration No. 34,111

600 13th Street, N.W.
Washington, DC 20005-3096
(202) 756-8000 KEG:tlb
Facsimile: (202) 756-8087
Date: March 1, 2004



62807-150
OTAKE
November 18, 2003

日 本 国 特 許 庁
JAPAN PATENT OFFICE

McDermott, Will & Emery

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日 2 0 0 3 年 3 月 5 日
Date of Application:

出 願 番 号 特 願 2 0 0 3 - 0 5 7 9 3 6
Application Number:
[ST. 10/C] : [J P 2 0 0 3 - 0 5 7 9 3 6]

出 願 人 株式会社日立製作所
Applicant(s):

2 0 0 3 年 1 0 月 2 8 日

特許庁長官
Commissioner,
Japan Patent Office

今 井 康 夫



【書類名】 特許願

【整理番号】 K02013801A

【あて先】 特許庁長官殿

【国際特許分類】 G06F 11/28

【発明者】

【住所又は居所】 神奈川県横浜市戸塚区戸塚町 5 0 3 0 番地 株式会社日立製作所 ソフトウェア事業部内

【氏名】 大竹 厚

【特許出願人】

【識別番号】 000005108

【氏名又は名称】 株式会社日立製作所

【代理人】

【識別番号】 100075096

【弁理士】

【氏名又は名称】 作田 康夫

【手数料の表示】

【予納台帳番号】 013088

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 プログラム管理方法及びシステム並びにその処理プログラムを格納した記憶媒体

【特許請求の範囲】

【請求項 1】 所定URLへアクセス要求を送付して、エラーが発生したとき、WSDLファイルの変更によるエラーか否かを判定し、WSDLファイルの変更によるエラーの場合は、上記プログラムの変更箇所を決定し、所定手続きに基づいて、上記変更箇所を変更して、上記プログラムを実行するWebアクセス方法および装置並びにその処理プログラム。

【請求項 2】 前記アクセス要求送付のエラーの検出は、エラーが発生した場合に受け取るfaultメッセージにより起動され、faultメッセージの内容を解析し、その種類に応じて、送付したアクセス要求メッセージのエラーかどうかを判断するように構成したことを特徴とする請求項 1 記載のプログラム管理方法。

【請求項 3】 前記プログラムの変更箇所の検出は、請求項 2 記載のアクセス要求送付エラー検出の判断により起動され、エラーが生じたアクセス要求メッセージの解析を行い、その結果により改訂前WSDLファイルと改訂後WSDLファイルを比較することにより、発生したエラーがWSDLファイルの変更によるエラーかどうかを判断し、その結果によりプログラムの変更箇所を検出するように構成したことを特徴とする請求項 1 記載のプログラム管理方法。

【請求項 4】 前記プログラムの変更箇所の変更は、請求項 3 記載の変更箇所検出の判断により起動され、プログラムの変更箇所を変更するように構成したことを特徴とする請求項 1 記載のプログラム管理方法。

【請求項 5】 前記プログラム変更箇所変更後の実行は、請求項 4 記載の変更箇所変更の判断により起動され、変更したプログラムを実行するように構成したことを特徴とする請求項 1 記載のプログラム管理方法。

【請求項 6】 前記アクセス要求送付のエラー検出は、受け取ったfaultメッセージの種類が、送付したアクセス要求メッセージのエラーであることを解析するように構成したことを特徴とした請求項 2 記載のプログラム管理方法。

【請求項 7】 前記プログラムの変更箇所の検出は、エラーが発生したアクセス要

求メッセージをプログラムから取得し、そのメッセージがプログラム作成時に取得保存した改訂前WSDLファイルにしたがって作成されているかを解析し、その結果により変更箇所を検出するように構成したことを特徴とした請求項3記載のプログラム管理方法。

【請求項8】前記プログラムの変更箇所の検出は、送付アクセス要求メッセージの正否確認により、改訂前WSDLファイルから生成された送付アクセス要求メッセージであることを確認できた場合に、プログラム作成時に取得保存した改訂前WSDLファイルの取得条件から改訂後WSDLファイルの保存場所情報を取得するように構成したことを特徴とした請求項3記載のプログラム管理方法。

【請求項9】前記プログラムの変更箇所の検出は、送付アクセス要求メッセージの正否確認の判断結果により取得した改訂後WSDLファイルの保存場所情報から、改訂後WSDLファイルを取得するように構成したことを特徴とした請求項3記載のプログラム管理方法。

【請求項10】前記プログラムの変更箇所の検出は、送付アクセス要求メッセージの正否確認の判断結果により取得した改訂後WSDLファイルの保存場所情報から、改訂後WSDLファイルを取得し、改訂前WSDLファイルと改訂後WSDLファイルを比較し、WSDLファイルの変更の種類を解析するように構成したことを特徴とした請求項3記載のプログラム管理方法。

【請求項11】前記プログラムの変更箇所の検出は、送付アクセス要求メッセージの正否確認の判断結果により取得した改訂後WSDLファイルの保存場所情報から、改訂後WSDLファイルを取得保存し、WSDLファイル変更の種類の結果から、プログラムの変更箇所を検出するように構成したことを特徴とした請求項3記載のプログラム管理方法。

【請求項12】前記プログラムの変更箇所の変更は、請求項3記載のプログラム変更箇所の検出により、検出された変更内容に従い、プログラムの変更箇所の生成を行うように構成したことを特徴とする請求項4記載のプログラム管理方法。

【請求項13】前記プログラムの変更箇所変更後の実行は、請求項12記載の変更により生成したプログラムを実行するように構成したことを特徴とする請求項5記載のプログラム管理方法。

【発明の詳細な説明】**【0 0 0 1】**

【発明の属する技術分野】 本発明は、プログラムの変更管理に関する。

【従来の技術】 近年、クライアント／サーバプログラムは分散オブジェクト技術を利用し、資源の有効利用、部品化が進んできている。インターネットを用い、ローカルなネットワークを超えた分散オブジェクト技術として、David A. Chappell他著「Java Web Services」(出版社：O'Reilly & Associates, Inc., 2002.3) (JavaおよびJava関連の商標およびロゴは、米国もしくはその他のSun Microsystems, Inc. の商標または登録商標です) に記載されているWebサービスがある。本文献では、Webサービスの基礎技術として3つの技術を挙げており、SOAP(Simple Object Access Protocol)、WSDL(Web Service Description Language)、UDDI(Universal Description Discovery, and Integration)について記述している。

【0 0 0 2】

本文献によれば、SOAPは構造化文書交換に使う封筒構造の定義であり、この定義にしたがって記述した構造化文書をSOAPメッセージと呼んでいる。Webサービスを用いたクライアント／サーバプログラムではこのSOAPメッセージを用いて通信を行う。このような技術は本文献 P. 25～P53に開示されている。

次にWSDLとは、Webサービスのインタフェースを定義するための構造化文書フォーマットである。Webサービスを用いたクライアント／サーバプログラムでは、このWSDLフォーマットにしたがい記述したWSDLファイルをインタフェースとし、SOAPメッセージを用い通信を行う。このような技術は本文献P. 72～P95に開示されている。

最後に、UDDIとは、Webサービスに対するグローバルレジストリと標準仕様を提供しており、インターネットで共通に利用できるレジストリとして公開されている。したがって、Webサービスを用いたクライアント／サーバプログラムでは、サーバプログラムのWebサービスをこのUDDIレジストリに登録しておき、クライアントプログラムはこのUDDIレジストリに登録されているWebサービスを見つけることにより、サーバプログラムのWebサービスが利用できるようになる。このような技術は本文献 P. 98～P139に開示されている。

【0003】

【非特許文献1】

David A. Chappell他著「Java Web Services」(出版社：O'Reilly & Associates, Inc., 2002.3)

【0004】

【発明が解決しようとする課題】

分散技術におけるアクセス要求送付先のインタフェース定義情報が変更されて、クライアントプログラムを変更し、作成する場合を考えてみる。保守作業者は、クライアントプログラムがエラーメッセージを受け付けた原因がインタフェース定義情報の変更によるものであることをわかっており、それをもとにクライアントプログラムを修正するものとする。

【0005】

上記のようなクライアントプログラムの保守作業に際しては、エラーを発生させた送付メッセージが、改定前のインタフェース定義情報にしたがって、生成されたメッセージかどうかを確認し、生成されたメッセージが正しい場合は、改訂後のインタフェース定義情報を取得し、改訂後のインタフェース定義情報にしたがったメッセージを送付するようにクライアントプログラムを修正する。

【0006】

しかしながら、従来の方式によるクライアントプログラムの生成は、インタフェース定義情報と雛型からクライアントプログラムを生成するものにすぎない。そのため、実行中のクライアントプログラムに発生したエラーがインタフェース定義情報の変更によるエラーであると原因がわかっているにもかかわらず、保守作業者は、一旦クライアントプログラムを停止させ、改訂後のインタフェース定義情報を取得し、それにそったクライアントプログラムの雛型の作成を再度行い、インタフェース定義情報の変更箇所とプログラムの修正個所の対応関係を判断しながら作業を行う手間のかかる作業となってしまう。

【0007】

本発明の目的は、実行中クライアントプログラムにおける、サーバプログラムの変更による修正と実行を動的に行うようにすることにある。

【0008】

【課題を解決するための手段】

前記課題を改善するため、クライアントプログラムが、サーバプログラムを提供している所定URLへアクセス要求を送付して、エラーが発生したとき、このエラーの発生原因を解析するためのプログラムを起動し、インタフェース定義情報の変更によるエラーか否かを判定し、インタフェース定義情報の変更によるエラーの場合は、上記クライアントプログラムの変更箇所を解析するプログラムを起動し、変更箇所の特定を行い、所定手続きに基づいて上記クライアントプログラムの上記変更箇所を変更して、上記クライアントプログラムを実行する。

【0009】

【発明の実施の形態】

以下、本発明の第一の実施例を図面に基づいて説明する。

図1は本実施例のシステム構成を示す。ネットワーク5に接続されたコンピュータ1はプログラム管理機能を備えたクライアントプログラムサーバであり、コンピュータ2はネットワークを介してクライアントプログラムサーバのプログラムを利用する一般ユーザである。ネットワークに接続されたコンピュータ3はwebサービスを提供するプログラムを格納したwebサービスサーバである。またコンピュータ4はwebサービスを登録管理するデータベースを格納したUDDIレジストリである。本実施例はサーバクライアント間の通信手段について特定するものではないが、インターネット上でのHTTPプロトコルによる通信を前提として説明する。すなわち、クライアントプログラムサーバ1はHTTPサーバとしての機能を備え、一般ユーザ2はwebブラウザを介してクライアントプログラムサーバ1と通信するものとする。webサービスサーバ3はHTTPサーバとしての機能を備え、クライアントプログラムサーバ1とSOAPメッセージを用いて通信するものとする。また、UDDIレジストリ4はHTTPサーバとしての機能を備え、クライアントプログラムサーバ1とSOAPメッセージを用いて通信するものとする。従ってクライアントプログラムサーバ1は、ディスプレイ6とキーボード等のデータ入力装置7と、CPU8と、メモリ9と、改定前後のWSDLファイルを蓄積するデータベース10から構成され、メモリ9には、変更管理プログラム72が保持される。この変更

管理プログラム 72 には、開発環境 70 と実行環境 71 とが含まれる。実行環境 71 には HTTP サーバプログラム 24 と、クライアントプログラム 25 と、クライアントプログラム実行プログラム 26 と、アクセス要求エラー検出プログラム 27 と、プログラムエラー発生箇所特定プログラム 28 が含まれる。開発環境 70 には、エラー回避クライアントプログラム生成プログラム 29 が含まれる。また、一般ユーザ 2 は、ディスプレイ 11 と、入力装置 12 と、CPU 13 と、メモリ 14 から構成され、メモリ 14 には web ブラウザプログラム 31 が保持される。web サービスサーバ 3 は、ディスプレイ 15 と、入力装置 16 と、CPU 17 と、メモリ 18 とデータベース 30 から構成され、メモリ 18 には、HTTP サーバプログラム 32 と、web サービスプログラム 33 が保持される。また、web サービスサーバ 3 は、改定前 WSDL ファイル 36 と改定後 WSDL ファイル 37 をデータベース 30 に保持している。UDDI レジストリ 4 は、ディスプレイ 19 と、入力装置 20 と、CPU 21 と、メモリ 22 と、データベース 23 から構成され、メモリ 22 には、HTTP サーバプログラム 34 と、UDDI プログラム 35 が保持される。また、UDDI レジストリ 4 は、データベース 68 に改訂後 WSDL ファイル保存場所情報 51 を保持している。

【0010】

図 2 は本実施例で実行されるクライアントプログラム 25 の構成を示す。図に示すように、制御機能 38、業務処理機能 39、表示機能 40、アダプタプログラム 41、SOAP クライアントプログラム 42、WSDL スタブプログラム 43 から構成される。

一般ユーザクライアント 2 がクライアントプログラム 25 にリクエストを送付し、クライアントプログラム 25 が web サービスサーバ 3 にアクセス要求を送付、処理結果を受け取り、一般ユーザクライアントに処理結果を表示する。

Web サービスサーバ 3 は住民基礎情報である住民の名前、住所、年齢等の情報を登録管理するサーバプログラムであり、アクセス要求を受け付けるメッセージとして SOAP メッセージを用い、インタフェース定義情報の記述には WSDL を用いる。WSDL を用いて記述した改訂前 WSDL ファイル 36 の message タグ部分を図 3 に示す。101 は message タグであり、アクセス要求を受け付けるメソッド名の定義

としてname属性にsetJukiListを指定している。1 0 2はpartタグであり、setJukiListメソッドの引数の定義としてname属性にNameを指定し、この型定義としてtype属性にstringを指定している。1 0 3、1 0 4も同様にstring型で、Address変数とAge変数を指定している。

【0 0 1 1】

図4に改訂後WSDLファイル37のmessageタグ部分を示す。改訂前WSDLファイル36に105のpartタグが付加され改訂されている。105はpartタグのname属性としてBirthdayを指定し、type属性にstringを指定している。

【0 0 1 2】

以下、webサービスサーバ3の管理者が住民基礎情報の内容充実化をはかるために、登録関数の引数一つを増やし、生年月日を追加することにしたものとして、説明を行う。したがって、UDDIレジストリに登録する自サービスのインタフェースである改訂前WSDLファイル36の保存場所情報を改訂後WSDLファイル37の保存場所情報に入れ替えを行った。この時の改定後WSDLファイル37のmessageタグ部分は図4に示す。

【0 0 1 3】

クライアントプログラム2.5の表示機能40は、図5に示すHTMLを記述した改訂前のファイルであり、201はhtmlタグであり、202はheadタグ、203はtitleタグであり「住民基礎情報入力」をwebブラウザタイトルに表示するように定義している。204はbodyタグであり、ここからwebブラウザに表示される内容であることを示している。したがって205の文字列「住民基礎情報入力」の記述でwebブラウザに文字列「住民基礎情報入力」が表示される。206は一般ユーザクライアント2が入力するFORMタグであり、httpリクエストのPOSTメソッド実行を定義している。また、このときのアクションタイプとして、SETJUKIを定義している。

【0 0 1 4】

207から209はこのFORMタグで送信される内容を定義している。テキスト入力項目として207のName、208のAddress、209のAgeと、このテキスト入力項目207、208、209の情報を送信するためのsubmitタグ210と、

テキスト入力項目の内容を消去するresetタグ 2 1 1 から構成される。この改訂前JSPファイルが一般ユーザクライアント 2 のwebブラウザに表示される画面を図 7 に示す。

【 0 0 1 5 】

改訂後のJSPファイルの内容を図 6 に示す。改訂前JSPファイルにテキスト入力項目 2 1 2 のBirthdayが付加され改訂されている。この改訂後JSPファイルが一般ユーザクライアント 2 のwebブラウザに表示される画面を図 8 に示す。

【 0 0 1 6 】

次にテキスト入力項目とWSDLファイルのpartタグとの関係を説明する。図 5 の改訂前JSPファイルのテキスト入力項目は、図 3 に示す改定前WSDLファイル 3 6 のmessageタグの子要素であるpartタグで記述された引数と同一の項目である。つまり、partタグで記述された 1 0 2 のName、1 0 3 のAddress、1 0 4 のAgeが図 5 のJSPファイルのテキスト入力項目である 2 0 7 のName、2 0 8 のAddress、2 0 9 のAgeとなる。

次に、このテキスト入力項目がクライアントプログラム 2 5 で処理される手順を説明する。

【 0 0 1 7 】

図 2 において一般ユーザクライアント 2 がwebブラウザプログラム 3 1 から入力したテキスト入力項目の内容は 2 5 __ 1 のHTTPリクエストでクライアントプログラム 2 5 へ送られ、制御機能 3 8 がリクエストを受け付ける。このリクエストは 2 5 __ 2 で業務処理機能 3 9 へ処理要求が行われる。2 5 __ 3 で業務処理機能 3 9 はアダプタプログラム 4 1 へ、リクエストを引数として処理を依頼する。アダプタプログラム 4 1 はリクエストのテキスト入力項目の内容を取り出し、SOAPクライアントプログラム 4 2 のメソッドを実行し、WSDLスタブプログラム 4 3 を使用し、webサービスサーバ 3 にアクセス要求を送付し、webサービスを利用する。webサービスサーバ 3 はアクセス要求を受け付けた場合、2 5 __ 4 で結果として戻り値を返却する。返却された戻り値は、WSDLスタブ 4 3、SOAPクライアントプログラム 4 2、アダプタプログラム 4 1 を通じて業務処理機能 3 9 に格納される。2 5 __ 6 で制御機能 3 8 は業務処理機能 3 9 の処理終了の通知を受けて、2

5__7の表示機能40に出力要求を行う。25__8で表示機能40では業務処理機能39に格納されている戻り値を取得し、25__9で結果を受け取る。25__9で受け取った結果をレスポンスとして一般ユーザクライアント2のwebブラウザプログラム31に表示する。

【0018】

次に、クライアントプログラム25で動作する各プログラムについて説明する。

WSDLスタブプログラム43は図3の改定前WSDLファイル36から生成される。図9に改定前WSDLファイル36から生成されたWSDLスタブプログラムのソースコードを示す。2010はWSDLスタブプログラムのパッケージ宣言である。ここでは、パッケージとしてjp.co.Ohtake.wsdlというパッケージ名称とし各クラスはこのパッケージに生成されることを意味する。2020はJukiList_Serviceのインタフェース宣言であり、メソッドとして2030のsetJukiListを持つことを宣言している。また、この2030では、このメソッドの引数としてString型のName、Address、Ageを持ち、戻り値としてboolean型を返すことを定義している。2040は2020の定義の終わりを表している。

【0019】

2050は2010と同様にパッケージ宣言である。2060はregistry.Registryクラスのインポート宣言をしている。2070ではregistry.RegistryExceptionクラスのインポート宣言をしている。2080から2150ではJukiList_ServiceHelperクラスを定義しており、2090と2120にこのクラスが持つメソッドを定義している。2100では2090で宣言されたメソッドbind()の戻り値を定義している。2110は2090の定義の終わりである。2130では2120で宣言されたメソッドbind(String url)の戻り値を定義している。

【0020】

図10は図4に示す改訂後WSDLファイル37から生成されるWSDLスタブプログラムのソースコードである。図9に示すソースコードとの違いは、2030に示すsetJukiListメソッドの引数にString型Birthdayが追加されたことである。これを図10の2160に示す。

【 0 0 2 1 】

次に、改訂前WSDLファイル 3 6 から生成された図 9 のWSDLスタブプログラムを利用するSOAPクライアントプログラム 4 2 のソースコードを図 1 1 に示す。3 0 1 0 はパッケージ宣言である。3 0 2 0 から 3 0 7 0 でInvoke_jukiListクラスを定義している。3 0 3 0 はこのInvoke_jukiListクラスのメソッドの宣言であり、引数としてString型のName、Address、Ageと戻り値としてboolean型を定義している。3 0 4 0 は 2 0 9 0 から 2 1 1 0 で定義されたJukiList_ServiceHelperクラスのメソッドbind()を用いてJukiList_Serviceクラスのインスタンスを得ている。次に 3 0 5 0 では、JukiList_Serviceクラスのインスタンスを用いて 2 0 3 0 で定義されたメソッドsetJukiListを使用し、結果を得ている。

【 0 0 2 2 】

図 1 2 に図 1 1 のSOAPクライアントプログラムを利用するアダプタプログラム 4 1 のソースコードを示す。アダプタプログラム 4 1 は 4 0 1 0 でjp.co.Ohtake.adapterパッケージ宣言をしている。4 0 2 0 ではjp.co.Ohtake.wsdlパッケージ全てをインポートし、4 0 3 0 ではjavax.servlet.httpパッケージ全てをインポートしている。4 0 4 0 から 4 1 5 0 では、adp_Invoke_jukiListクラスの定義を行っている。4 0 5 0 から 4 1 4 0 ではこのクラスのメソッド定義をしている。このメソッドadp_setjukiは戻り値がBoolean型で引数がHttpServletRequest型requestである。

【 0 0 2 3 】

4 0 6 0 では戻り値用の変数adp_rtnの初期化をfalseで行っている。次に 4 0 7 0 から 4 0 9 0 ではrequest変数が保持している値をそれぞれ取り出し、adp_name、adp_addreess、adp_ageへと代入している。4 1 0 0 では 3 0 2 0 で定義されたクラスのインスタンスを生成している。4 1 2 0 ではこのインスタンスでメソッドsetjukiを実行し、結果をadp_rtnに代入している。

【 0 0 2 4 】

図 1 3 は図 1 0 に示す改訂後WSDLファイル 3 7 から生成されるWSDLスタブプログラムのソースコードを利用するSOAPクライアントプログラムのソースコードである。図 1 1 との違いは、3 0 8 0 のメソッドの引数にString型のBirthdayが増

加している事と、3 0 9 0 のメソッドの引数にString型のBirthdayが増加している事である。

【0 0 2 5】

図 1 4 は図 1 3 に示すSOAPクライアントプログラムを利用するためのアダプタプログラムのソースコードである。図 1 2 との違いは、4 1 6 0 と 4 1 7 0 である。4 1 6 0 では、String型の変数adp_birthdayにrequest変数から値を取り出し、代入している。4 1 7 0 では、実行するメソッドsetjukiの引数adp_birthdayが増加している。

次に、webサービスサーバ3のwebサービスプログラム33のインタフェースが、改訂前WSDLファイル36から改訂後WSDLファイル37へ変更された場合における変更管理プログラム72の処理概要を図15に示す。

【0 0 2 6】

Faultメッセージ44を、クライアントプログラム25が受け取り、アクセス要求エラーの検出プログラム27が実行され、プログラムエラー発生個所特定プログラム28が実行され、エラー回避クライアントプログラム生成プログラム29が実行され、クライアントプログラム実行プログラム26が実行される。

【0 0 2 7】

クライアントプログラム25が、改訂後webサービスプログラム33を有するwebサービスサーバ3にアクセス要求を送付した結果、Faultメッセージ44を取得した時、クライアントプログラム25では、WSDLスタブプログラム43がFaultメッセージ44を取得し、SOAPクライアント42へ返信する。SOAPクライアント42はアダプタプログラム41へFaultメッセージ44を返信する。アダプタプログラム41がFaultメッセージ44を受け取ると、Faultメッセージ取得通知プログラム45を実行する。Faultメッセージ取得通知プログラム45を実行したアダプタプログラム41は完全停止フラグまたは、一時停止フラグに基づき処理結果を業務処理機能39に返却する。アダプタプログラム41から結果フラグを受け取った業務処理機能39は処理終了を制御機能38に伝える。処理終了を伝えられた制御機能38は表示機能40に出力要求を行う。出力要求を受けた表示機能40は業務処理機能39の処理結果を取得し、一般ユーザクライアント2

のwebブラウザプログラムにレスポンスを返す。

【0 0 2 8】

図 1 5 に示した処理概要の詳細を以下に示す。

まず、アクセス要求エラー検出プログラム 2 7 の手順を図 1 6 に示す。アクセス要求エラー検出プログラム 2 7 は、Fault メッセージ取得通知プログラム 4 5、Fault メッセージ解析プログラム 4 6 から構成される。

Fault メッセージ取得通知プログラム 4 5 の処理フローを図 1 9 に示す。Fault メッセージ取得通知プログラム 4 5 は Fault メッセージを受け取る (3 0 0)。次に Fault メッセージ解析プログラム 4 6 を実行し、Fault メッセージ 4 4 を送る (3 0 1)。Fault メッセージ解析プログラム 4 6 の戻り値を判定する (3 0 2)。Fault メッセージ解析プログラム 4 6 の戻り値が Client フラグの場合、アダプタプログラム 4 1 へ一時停止フラグを返信し (3 0 3)、Client フラグ以外の場合、完全停止フラグをアダプタプログラム 4 1 へ返信する (3 0 4)。

Fault メッセージ解析プログラム 4 6 の処理フローを図 2 0 に示す。Fault メッセージ解析プログラム 4 6 は受け取った Fault メッセージ 4 4 を解析する (4 0 0)。このメッセージに記述されている全てのタグについて以下の判定を行う (4 0 1)。

全てのタグの中から faultCode タグを見つけ出し (4 0 2)、このタグの値が Client エラーを表すエラーコードかどうかを判断する (4 0 3)。Client エラーを表すエラーコードの場合、既存メッセージ確認プログラム 4 7 を実行する (4 0 4)。既存メッセージ確認プログラム 4 7 の戻り値が Client フラグの場合 (4 0 5)、Client フラグを Fault メッセージ取得通知プログラム 4 5 に返信し (4 0 6)、Client フラグ以外の場合、Client フラグ以外を返信する (4 0 7)。4 0 3 で Client エラーを表すエラーコードでない場合、Fault メッセージ取得通知プログラム 4 5 へ Client フラグ以外を返信する (4 0 8)。

プログラムエラー発生箇所検出プログラム 2 8 の手順を図 1 7 に示す。プログラムエラー発生箇所検出プログラム 2 8 は、既存メッセージ確認プログラム 4 7、UDDI 検索プログラム 4 8、WSDL ファイル取得プログラム 5 2、WSDL ファイル比較プログラム 5 3 から構成される。

【 0 0 2 9 】

既存メッセージ確認プログラム 4 7 の処理フローを図 2 1 に示す。既存メッセージ確認プログラム 4 7 は、web サービスサーバ 3 にアクセス要求を送付したりリクエストメッセージ 4 9 を受け取る (5 0 0) 。次に、改訂前 WSDL ファイル 3 6 をデータベース 1 0 から取得し (5 0 1) 、改訂前 WSDL ファイル 3 6 から確認用リクエストメッセージ 6 7 を生成する (5 0 2) 。リクエストメッセージ 4 9 の全てのタグについて (5 0 3) 、生成した確認用リクエストメッセージ 6 7 のタグと同一かどうか判定する (5 0 4) 。

改訂前 WSDL ファイル 3 6 から生成されたメッセージである場合、UDDI 検索プログラム 4 8 を実行する (5 0 5) 。改訂前の WSDL ファイル 3 6 から生成されたメッセージではない場合、改訂前の WSDL ファイル 3 6 を WSDL スタブ生成プログラム 5 4 へ送る (5 0 9) 。5 0 5 で実行した UDDI 検索プログラム 4 8 の戻り値を判定する (5 0 6) 。戻り値が Client フラグの場合、Fault メッセージ解析プログラム 4 6 へ Client フラグを返信し (5 0 7) 、Client フラグ以外の場合、Client フラグ以外を返信する (5 0 8) 。

UDDI 検索プログラム 4 8 の処理フローを図 2 2 に示す。UDDI 検索プログラム 4 8 は改訂前 web サービス取得条件 5 0 をデータベース 1 0 から取得し (6 0 0) 、この条件に基づき、UDDI レジストリ 4 で web サービスの検索を行う。全ての web サービスについて検索し (6 0 1) 、改訂前 web サービス取得条件 5 0 に一致する web サービスを見つけることができた場合 (6 0 2) 、WSDL ファイル保存場所情報が改訂されているか判定する (6 0 3) 。

【 0 0 3 0 】

WSDL ファイル保存場所情報が改訂されている場合、UDDI レジストリ 4 に登録されている改訂後 WSDL ファイル保存場所情報 4 9 を取得し (6 0 4) 、WSDL ファイル取得プログラム 5 2 へ送る。WSDL ファイル取得プログラム 5 2 の戻り値を判定し (6 0 5) 、Client フラグの場合、既存メッセージ確認プログラム 4 7 へ Client フラグを返信し (6 0 6) 、Client フラグ以外の場合、Client フラグ以外を返信する (6 0 7) 。6 0 3 で WSDL ファイル保存場所情報が改訂されていない場合、既存メッセージ確認プログラム 4 7 へ WSDL 改訂なしエラーを返信する (6 0 8)

）。6 0 2 で改訂前webサービス取得条件 5 0 に一致するwebサービスを見つけることができなかった場合、既存メッセージ確認プログラム 4 7 へwebサービス未検出エラーを返信する（6 0 9）。

【0 0 3 1】

WSDLファイル取得プログラム 5 2 の処理フローを図 2 3 に示す。WSDLファイル取得プログラム 5 2 は、改訂後WSDLファイル保存場所情報 5 1 を受け取り（7 0 0）、改訂後WSDLファイル保存場所情報 5 1 にしたがひ、改訂後WSDLファイル 3 7 を取得する（7 0 1）。次に、改訂後WSDLファイル 3 7 が取得できたかどうかを判定し（7 0 2）、取得できた場合、取得した改訂後WSDLファイル 3 7 をWSDLファイル比較プログラム 5 3 へ送り、実行する（7 0 3）。実行したWSDLファイル比較プログラム 5 3 の戻り値について判定し（7 0 4）、戻り値がpartタグ変更である場合、UDDI検索プログラム 4 8 へClientフラグを返信し（7 0 5）、partタグ変更以外の場合、Clientフラグ以外を返信する（7 0 6）。7 0 2 で改訂後WSDLファイル 3 7 を取得できなかった場合、UDDI検索プログラム 4 8 へ、取得失敗エラーを返す（7 0 7）。

【0 0 3 2】

WSDLファイル比較プログラム 5 3 の処理フローを図 2 4 に示す。初めに、WSDLファイル比較プログラム 5 3 は改定後WSDLファイル 3 7 を取得し（8 0 0）、次に改定前WSDLファイル 3 6 を取得する（8 0 1）。

取得した改定前後のWSDLファイルの比較を行い（8 0 2）、抽出された相違部分がWSDLファイル中のmessageタグの子要素であるpartタグの増減であれば、WSDLスタブ生成プログラム 5 4 を実行し、改定後のWSDLファイル 3 7 を送る（8 0 3）。その後、WSDLファイル取得プログラム 5 2 へpartタグ変更を返信する（8 0 4）。8 0 2 での判定がpartタグ増減以外の変更の場合、partタグ以外変更エラーをWSDLファイル取得プログラム 5 2 へ返信する（8 0 5）。

【0 0 3 3】

図 1 8 に示すエラー回避クライアントプログラム生成プログラム 2 9 の手順を以下に示す。エラー回避クライアントプログラム生成プログラム 2 9 は、WSDLスタブ生成プログラム 5 4、SOAPクライアント生成プログラム 5 6、アダプタ生成

プログラム 5 8、JSP生成プログラム 6 0、クライアントプログラム生成プログラム 6 2 から構成される。

WSDLスタブ生成プログラム 5 4 の処理フローを図 2 5 に示す。WSDLスタブ生成プログラム 5 4 は改定後WSDLファイル 3 7 を受け取り（9 0 0）、WSDLスタブソース 5 5 を生成するコマンドを実行する（9 0 1）。このWSDLスタブソース 5 5 をSOAPクライアント生成プログラム 5 6 へ送り実行する（9 0 2）。次に、JSP生成プログラム 6 0 へ改訂後WSDLファイル 3 7 を送り、実行する（9 0 3）。

【0 0 3 4】

次に、SOAPクライアント生成プログラム 5 6 の処理フローを図 2 6 に示す。SOAPクライアント生成プログラム 5 6 はWSDLスタブソース 5 5 を受け取る（1 0 0 0）。次に、WSDLスタブソース 5 5 を用いて、SOAPクライアントプログラムソース 5 7 を生成する（1 0 0 1）。SOAPクライアントプログラムソース 5 7 の生成の詳細は 1 0 0 2 から 1 0 0 7 で行われ、生成結果を図 1 3 に示す。生成したSOAPクライアントプログラムソース 5 7 をアダプタ生成プログラム 5 8 へ送る（1 0 0 8）。

【0 0 3 5】

次に、アダプタ生成プログラム 5 8 の処理フローを図 2 7 に示す。アダプタ生成プログラム 5 8 は受け取ったSOAPクライアントプログラムソース 5 7 から（1 1 0 0）、アダプタプログラムソース 5 9 を生成する（1 1 0 1）。アダプタプログラムソース 5 9 の生成は 1 1 0 2 から 1 1 0 8 で行われ、生成した結果を図 1 4 に示す。次に図 2 5 の 9 0 3 で実行されたJSP生成プログラム 6 0 の処理が終了しているかどうかを判断し（1 1 0 9）、アダプタプログラムソース 5 9 をクライアントプログラム生成プログラム 6 2 へ送り実行する（1 1 1 0）。

【0 0 3 6】

JSP生成プログラム 6 0 の処理フローを図 2 8 に示す。JSP生成プログラム 6 0 は改訂後WSDLファイル 3 7 を受け取る（1 2 0 0）。この改訂後WSDLファイル 3 7 のmessageタグの引数部分を入力項目とするJSPファイル 6 1 を生成する（1 2 0 1）。1 2 0 2 から 1 2 0 4 にJSPファイル生成の詳細を示す。生成されたJSPファイル 6 1 の内容を図 6 に示す。次に、生成したJSPファイル 6 1 をクライア

ントプログラム生成プログラム 62 へ送り、実行する (1205)。

【0037】

クライアントプログラム生成プログラム 62 の処理を以下に説明する。クライアントプログラム生成プログラム 62 は、WSDL スタブソース 55 と SOAP クライアントプログラムソース 57 とアダプタソース 59 と JSP ファイル 61 を受け取る。次に受け取った WSDL スタブソース 55 と SOAP クライアントプログラムソース 57 とアダプタソース 59 を javac コマンドによりコンパイルし、class ファイルを生成する。生成された各 class ファイルを jar コマンドにより jar ファイル形式のファイル JAR ファイル 68 にする。次に、JAR ファイル 68 と JSP ファイル 61 を jar コマンドの -war オプションにて war ファイル形式のファイル WAR ファイル 73 にする。最後にこの WAR ファイル 73 をクライアントプログラム実行プログラム 26 へ送り、クライアントプログラム実行プログラム 26 を実行する。

【0038】

クライアントプログラム実行プログラム 26 は、改定前の WSDL ファイル 36 に基づいた WAR ファイルを使用したクライアントプログラム 25 を停止し、クライアントプログラム生成プログラム 62 から受け取った WAR ファイル 73 を使用したクライアントプログラムを配置し、実行する。

【0039】

【発明の効果】

本発明によれば、実行中のクライアントプログラムについて、インタフェースの変更による対応変更管理が動的に行えるため、サービス停止期間を短縮することが可能となる。

【図面の簡単な説明】

【図 1】 本発明を適用したシステム構成を示す図である。

【図 2】 本発明に適用されるクライアントプログラムの構成を示す図である。

【図 3】 本発明に適用される改定前 WSDL ファイルのメッセージ定義を示す図である。

【図 4】 本発明に適用される改定後 WSDL ファイルのメッセージ定義を示す図である。

【図 5】本発明に適用されるクライアントプログラムの改訂前の表示機能を示す図である。

【図 6】本発明に適用されるクライアントプログラムの改訂後の表示機能を示す図である。

【図 7】本発明に適用される改訂前の一般ユーザクライアントに表示される画面を示す図である。

【図 8】本発明に適用される改訂後の一般ユーザクライアントに表示される画面を示す図である。

【図 9】本発明に適用される改訂前のWSDLスタブプログラムコードを示す図である。

【図 10】本発明に適用される改訂後のWSDLスタブプログラムコードを示す図である。

【図 11】本発明に適用される改訂前のSOAPクライアントプログラムコードを示す図である。

【図 12】本発明に適用される改訂前のアダプタプログラムコードを示す図である。

【図 13】本発明に適用される改訂後のSOAPクライアントプログラムコードを示す図である。

【図 14】本発明に適用される改訂後のアダプタプログラムコードを示す図である。

【図 15】本発明の実行形態の概略を示す図である。

【図 16】アクセス要求エラーを検出し、Faultメッセージの解析ステップを示す図である。

【図 17】改定後WSDLファイルを取得し、プログラムエラー発生箇所検出ステップを示す図である。

【図 18】プログラムエラー発生箇所検出ステップで検出された変更部分について、クライアントプログラムを生成するエラー回避クライアントプログラムの生成ステップを示す図である。

【図 19】本発明に適用されるFaultメッセージ取得通知プログラムの処理フロ

ーを示す図である。

【図 2 0】本発明に適用されるFaultメッセージ解析プログラムの処理フローを示す図である。

【図 2 1】本発明に適用される既存メッセージ確認プログラムの処理フローを示す図である。

【図 2 2】本発明に適用されるUDDI検索プログラムの処理フローを示す図である。

【図 2 3】本発明に適用されるWSDLファイル取得プログラムの処理フローを示す図である。

【図 2 4】本発明に適用されるWSDLファイル比較プログラムの処理フローを示す図である。

【図 2 5】本発明に適用されるWSDLスタブ生成処理プログラムの処理フローを示す図である。

【図 2 6】本発明に適用されるSOAPクライアント生成プログラムの処理フローを示す図である。

【図 2 7】本発明に適用されるアダプタ生成プログラムの処理フローを示す図である。

【図 2 8】本発明に適用されるJSP生成プログラムの処理フローを示す図である。

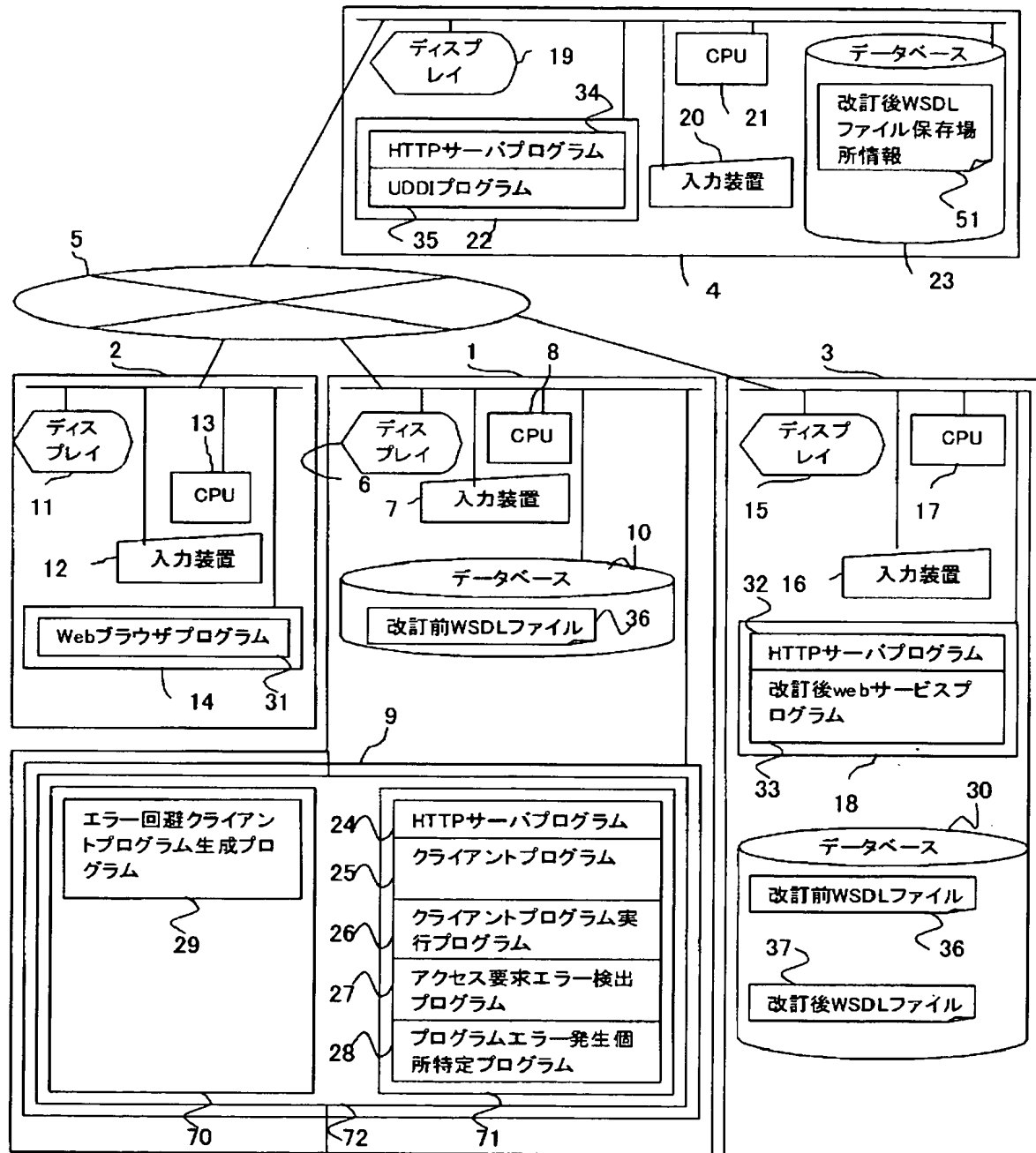
【符号の説明】

- 1…クライアントプログラム変更管理サーバ
- 2…一般ユーザクライアント
- 3…webサービスサーバ
- 4…UDDIレジストリサーバ
- 5…ネットワーク

【書類名】 図面

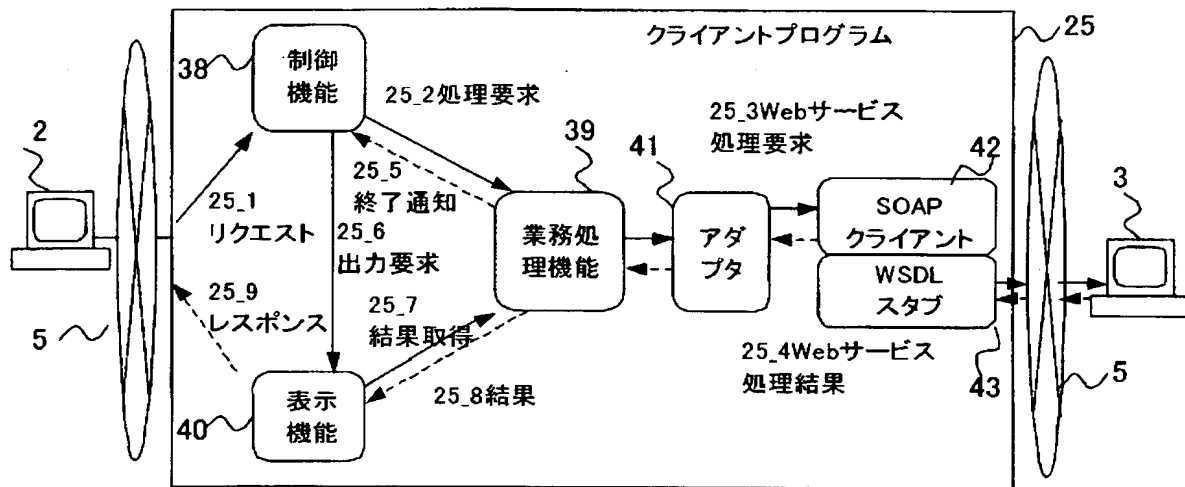
【図1】

図1



【図 2】

図 2



【図 3】

図 3

```

:
<message name="setJukiList"> 101
  <part name="Name" type="xsd:string"/> 102
  <part name="Address" type="xsd:string"/> 103
  <part name="Age" type="xsd:string"/>
</message> 104
:

```

【図 4】

図 4

```

:
<message name="setJukiList"> 101
  <part name="Name" type="xsd:string"/> 102
  <part name="Address" type="xsd:string"/> 103
  <part name="Age" type="xsd:string"/> 104
  <part name="Birthday" type="xsd:string"/>
</message> 105
:

```

【図 5】

図 5

```
<html> 201
<head> 202
<title>住民基礎情報入力</title> 203
</head>
<body> 204
<center>住民基礎情報入力 205
<form method="POST" action="SETJUKI"> 206
  <p>Name<input type="text" name="Name" size="100"></p> 207
  <p>Address<input type="text" name="Address" size="100"></p> 208
  <p>Age<input type="text" name="Age" size="100"></p> 209
  <p><input type="submit" value="送信" name="B1"> 210
    <input type="reset" value="リセット" name="B2"></p> 211
</form>
</center>
</body>
</html>
```

【図 6】

図 6

```
<html> 201
<head> 202
<title>住民基礎情報入力</title> 203
</head>
<body> 204
<center>住民基礎情報入力 205
<form method="POST" action="SETJUKI"> 206
  <p>Name<input type="text" name="Name" size="100"></p> 207
  <p>Address<input type="text" name="Address" size="100"></p> 208
  <p>Age<input type="text" name="Age" size="100"></p> 209
  <p>Birthday<input type="text" name="Birthday" size="100"></p> 212
  <p><input type="submit" value="送信" name="B1"> 210
    <input type="reset" value="リセット" name="B2"></p> 211
</form>
</center>
</body>
</html>
```


【図 7】

図 7

住民基礎情報入力 ✓ 205

Name 207

Address 208

Age 209

210 211

【図 8】

図 8

住民基礎情報入力 ✓ 205

Name 207

Address 208

Age 209

Birthday 212

210 211

【図 9】

図 9

```
2010 package jp.co.Ohtake.wsdl;

2020 public interface JukiList_Service {
2030     boolean setJukiList(String Name,String Address,String Age);
2040 }

2050 package jp.co.Ohtake.wsdl;

2060 import registry.Registry;
2070 import registry.RegistryException;

2080 public class JukiList_ServiceHelper{
2090     public static JukiList_Service bind() throws RegistryException{
2100         return bind("http://www.ohtake_service/wsdl/JukiListService.wsdl");
2110     }
2120     public static JukiList_Service bind(String url) throws RegistryException{
2130         return (JukiList_Service)Registry.bind( url, JukiList_Service.class);
2140     }
2150 }
```

【図 1 0】

図 1 0

```
2010 package jp.co.Ohtake.wsdl;

2020 public interface JukiList_Service {
2160   boolean setJukiList(String Name,String Address,String Age,String Birthday);
2040 }

2050 package jp.co.Ohtake.wsdl.;

2060 import registry.Registry;
2070 import registry.RegistryException;

2080 public class JukiList_ServiceHelper{
2090   public static JukiList_Service bind() throws RegistryException{
2100       return bind("http://www.ohtake_service/wsdl/JukiListService.wsdl");
2110   }
2120   public static JukiList_Service bind(String url) throws RegistryException{
2130       return (JukiList_Service)Registry.bind( url, JukiList_Service.class);
2140   }
2150 }
```

【図 1 1】

図 1 1

```
3010 package jp.co.Ohtake.wsdl;

3020 public class Invoke_jukiList{
3030   public boolean setjuki(String Name, String Address, String Age ) throws
                                     Exception{
3040       JukiList_Service jukiListService = JukiList_ServiceHelper.bind();
3050       boolean rtn = jukiListService.setJukiList( Name, Address, Age );
3060   }
3070 }
```

【図 1 2】

図 1 2

```
4010 package jp.co.Ohtake.adapter;

4020 import jp.co.Ohtake.wsdl.*;
4030 import javax.servlet.http.*;

4040 public class adp_Invoke_jukiList{
4050     public Boolean adp_setjuki( HttpServletRequest request )throws Exception{
4060         boolean adp_rtn = false;
4070         String adp_name=request.getAttribute( "Name" );
4080         String adp_address=request.getAttribute( "Address" );
4090         String adp_age=request.getAttribute( "Age" );

4100         Invoke_jukiList soapclient = new Invoke_jukiList();
4120         adp_rtn = soapclient.setjuki(adp_name, adp_address, adp_age );
4130         return adp_rtn;
4140 }
4150 }
```

【図 1 3】

図 1 3

```
3010 package jp.co.Ohtake.wsdl.

3020 public class Invoke_jukiList{
3080     public boolean setjuki(String Name,String Address,String Age,String
    Birthday )    throws Exception{
3040         JukiList_Service jukiListService = JukiList_ServiceHelper.bind();
3090         boolean rtn =jukiListService.setJukiList( Name, Address, Age, Birthday );
3060     }
3070 }
```

【図 1 4】

図 1 4

```
4010 package jp.co.Ohtake.adapter;

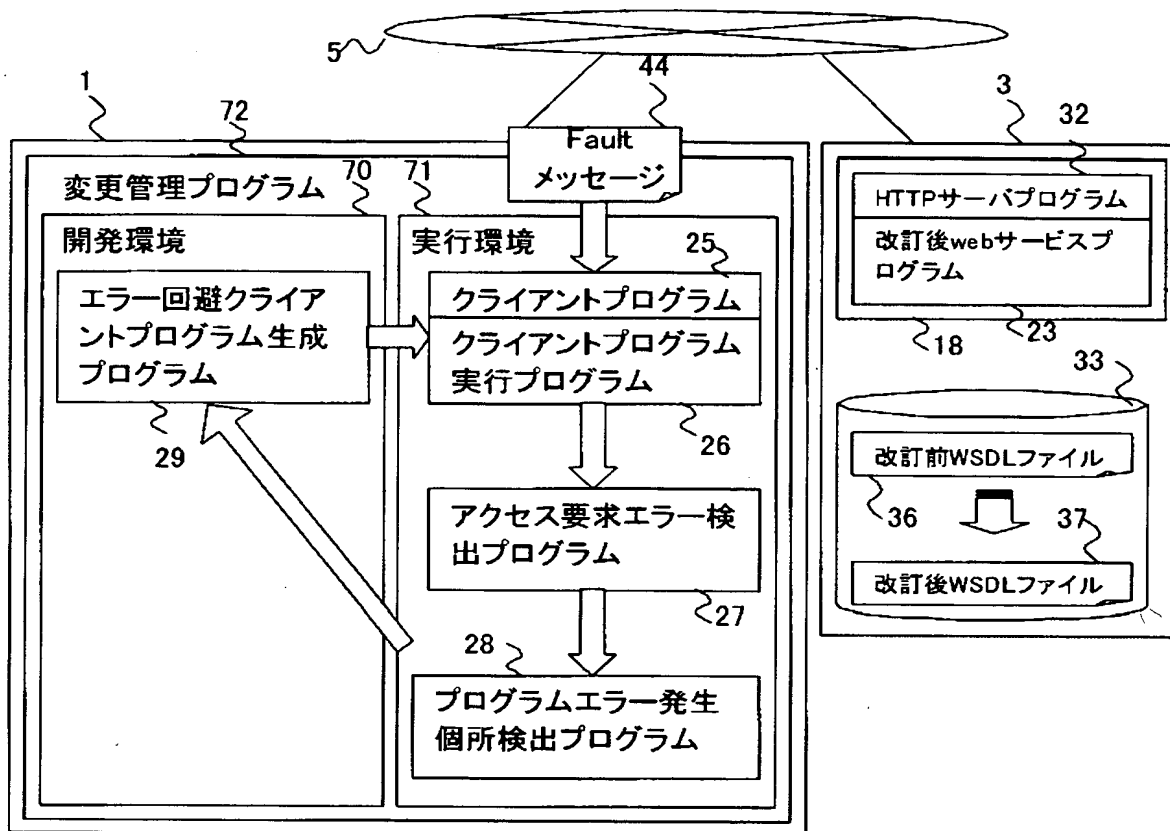
4020 import jp.co.Ohtake.wsdl.*;
4030 import javax.servlet.http.*;

4040 public class adp_Invoke_jukiList{
4050     public Boolean adp_setjuki( HttpServletRequest request )throws Exception{
4060         boolean adp_rtn = false;
4070         String adp_name=request.getAttribute( "Name" );
4080         String adp_address=request.getAttribute( "Address" );
4090         String adp_age=request.getAttribute( "Age" );
4160         String adp_birthday=request.getAttribute( "Birthday" );

4100         Invoke_jukiList soapclient = new Invoke_jukiList();
4170         adp_rtn=soapclient.setjuki(adp_name,adp_address,adp_age,
adp_birthday);
4140     }
4150 }
```

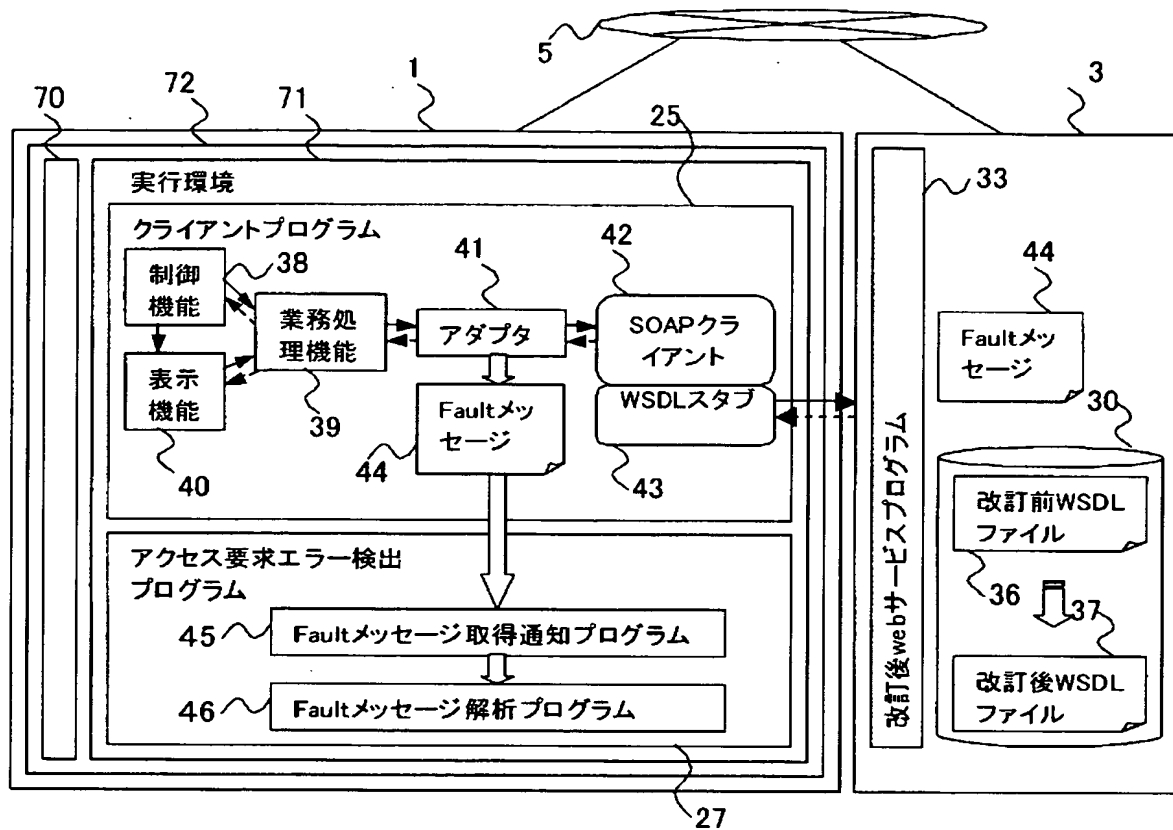
【図15】

図15



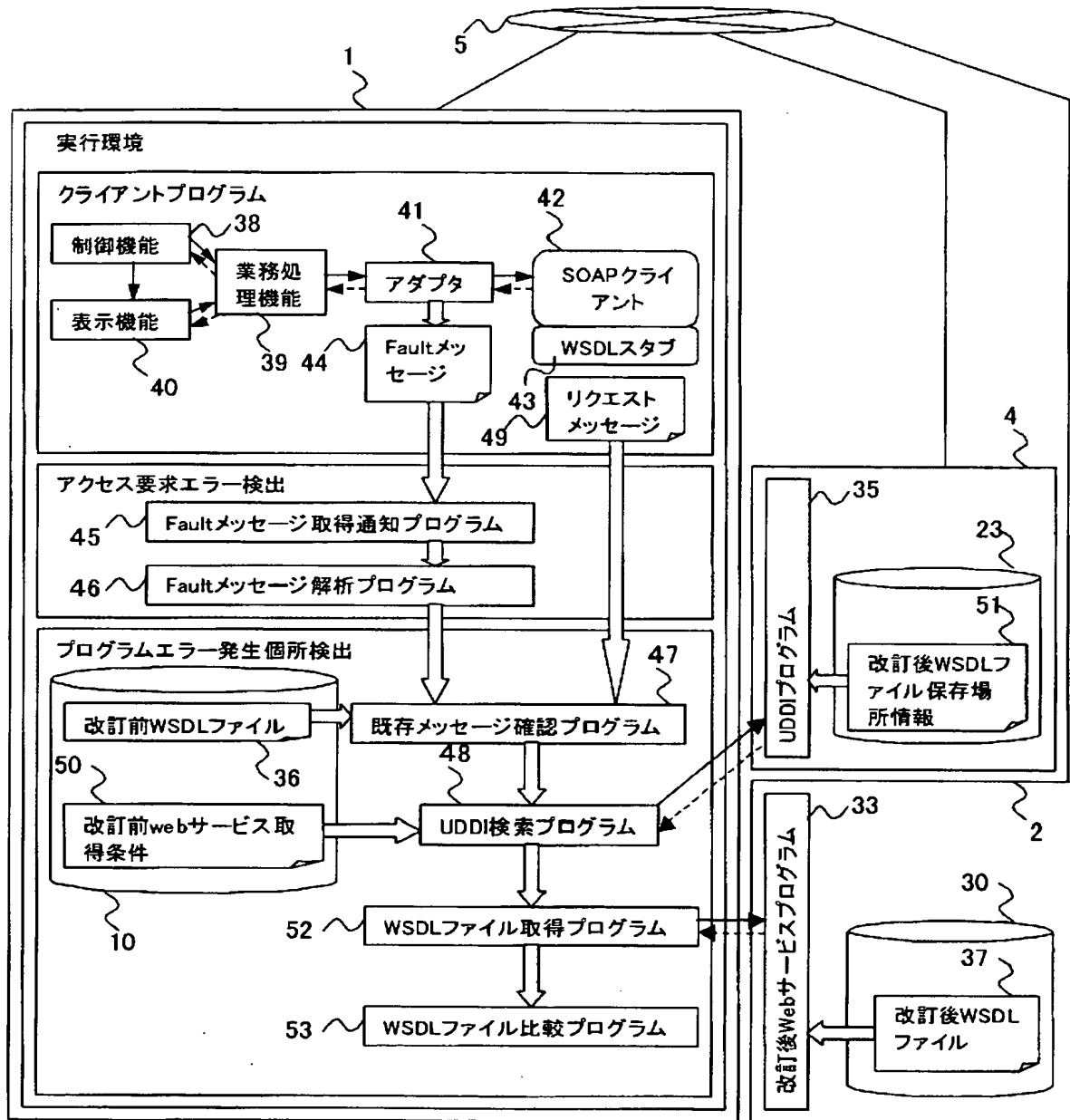
【図 16】

図 16



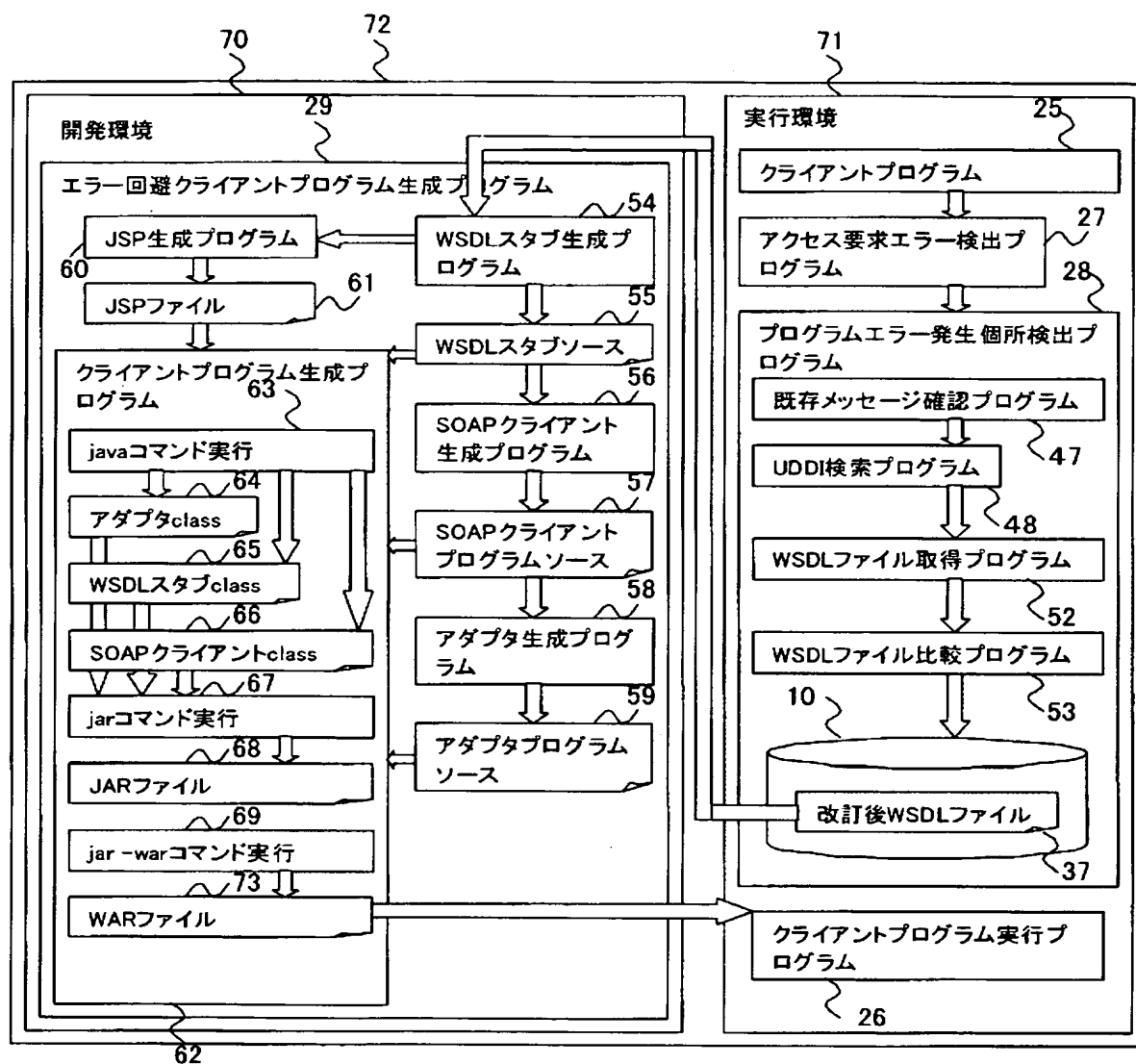
【図 17】

図 17



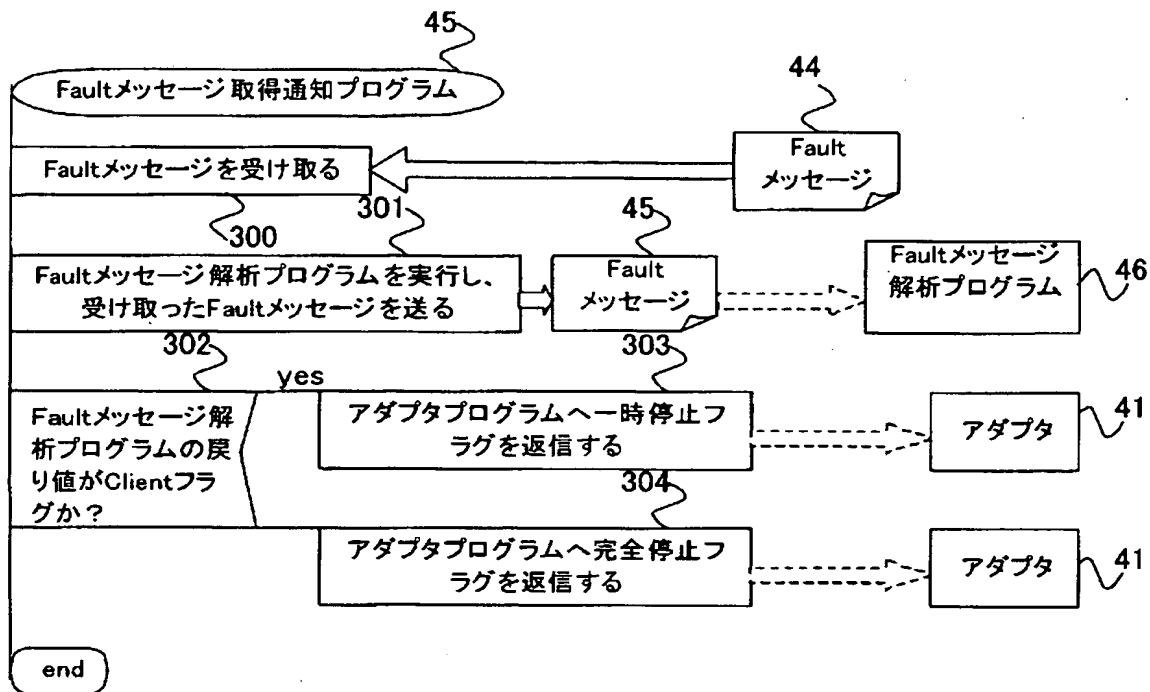
【図18】

図18



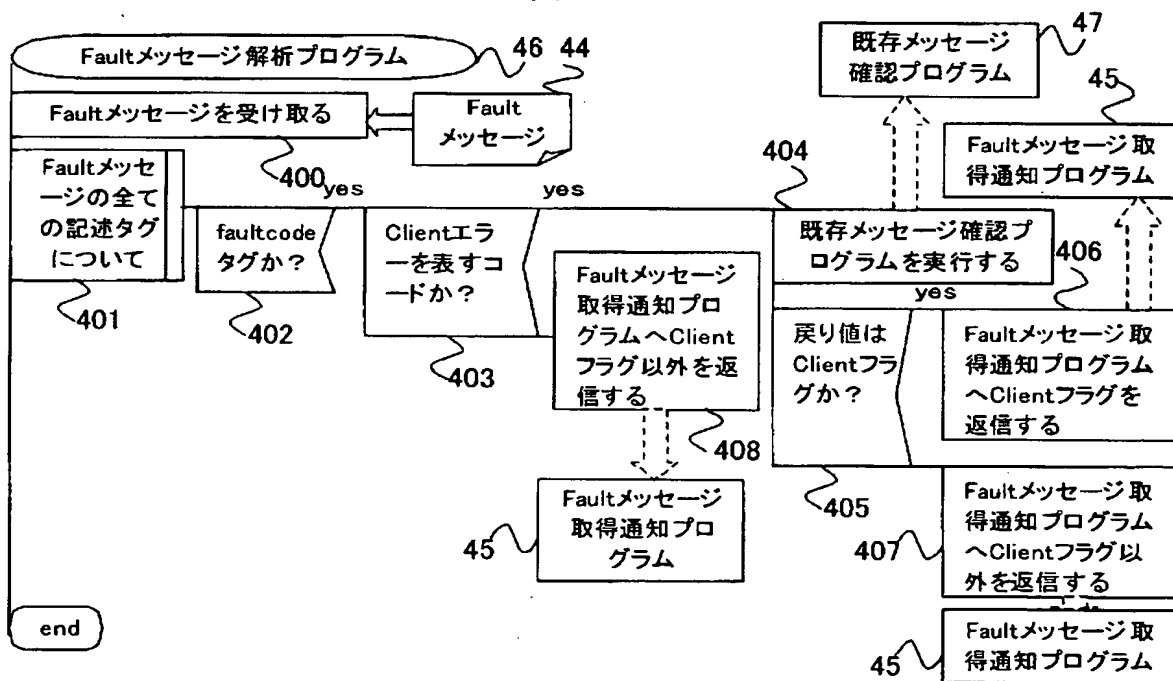
【図 19】

図 19



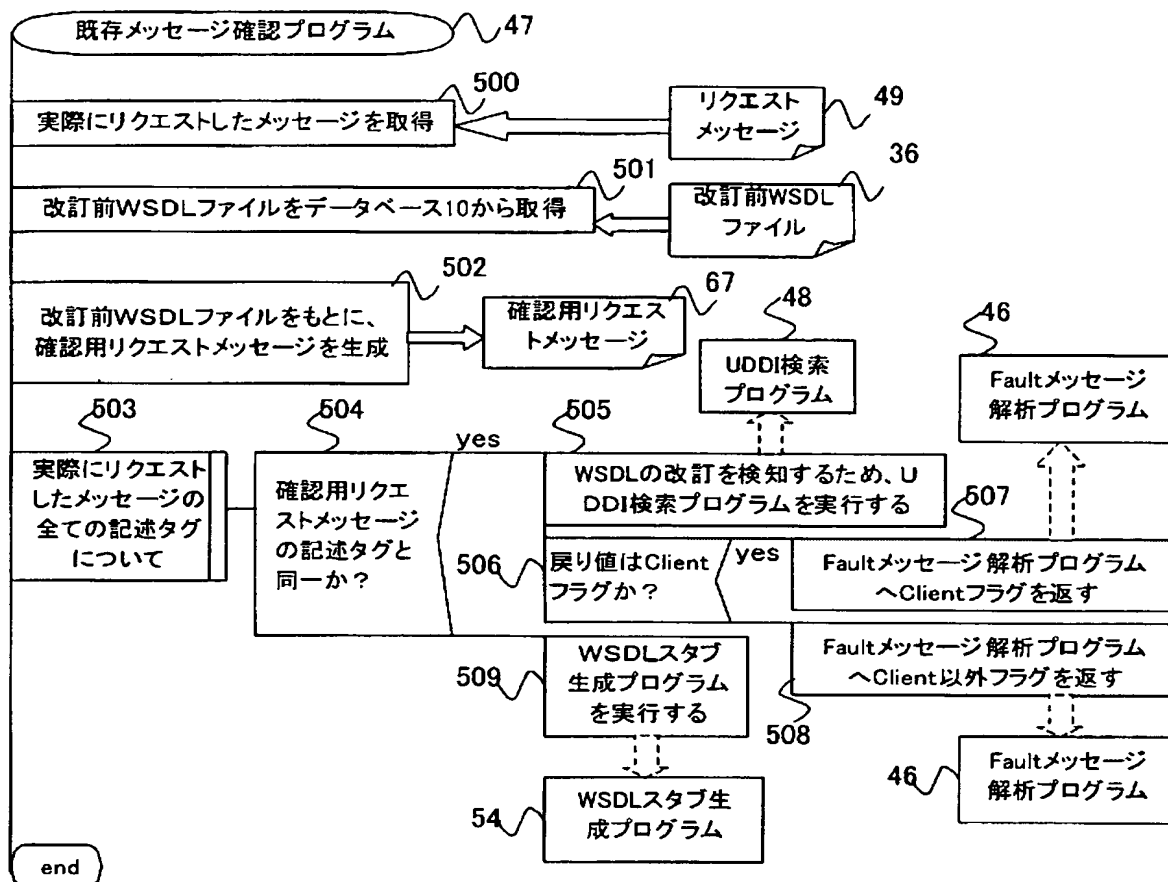
【図 20】

図 20

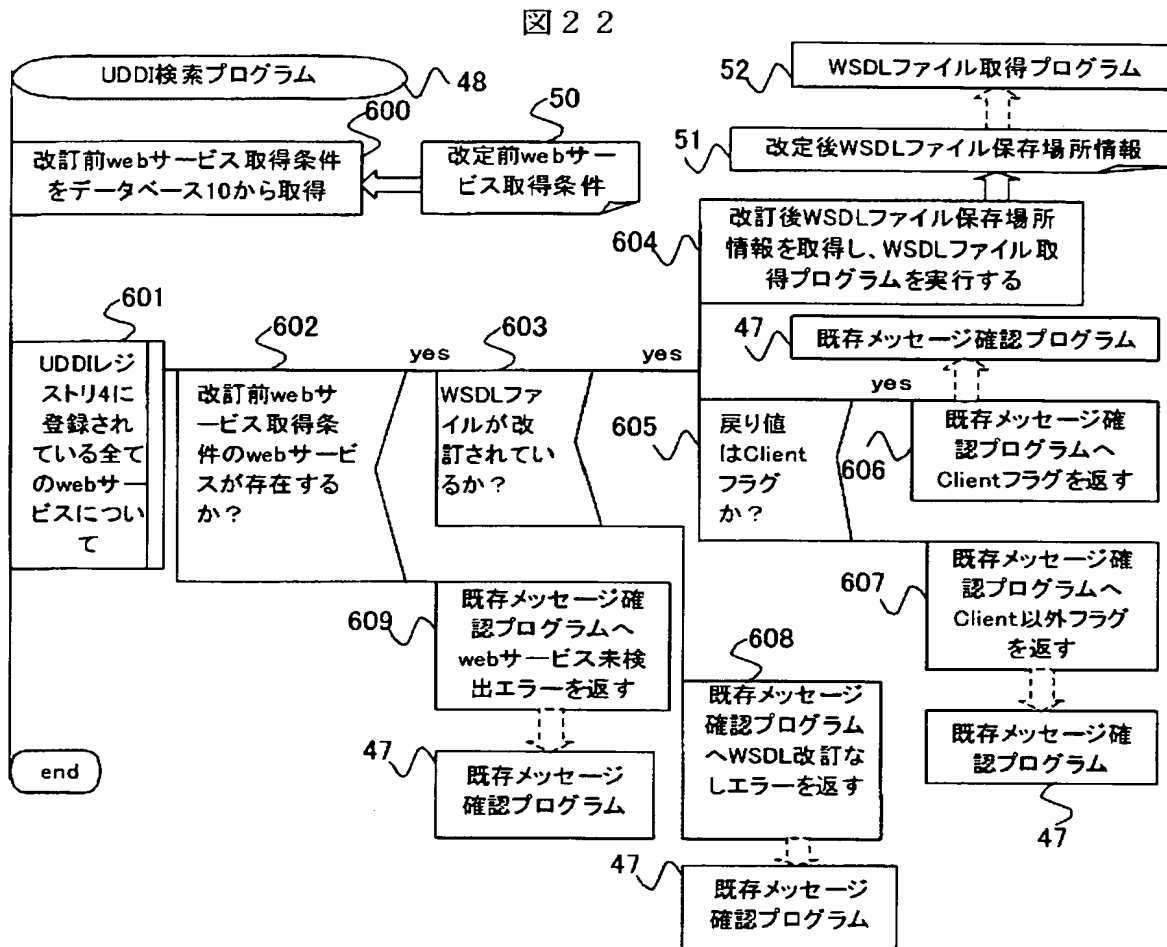


【図 21】

図 21

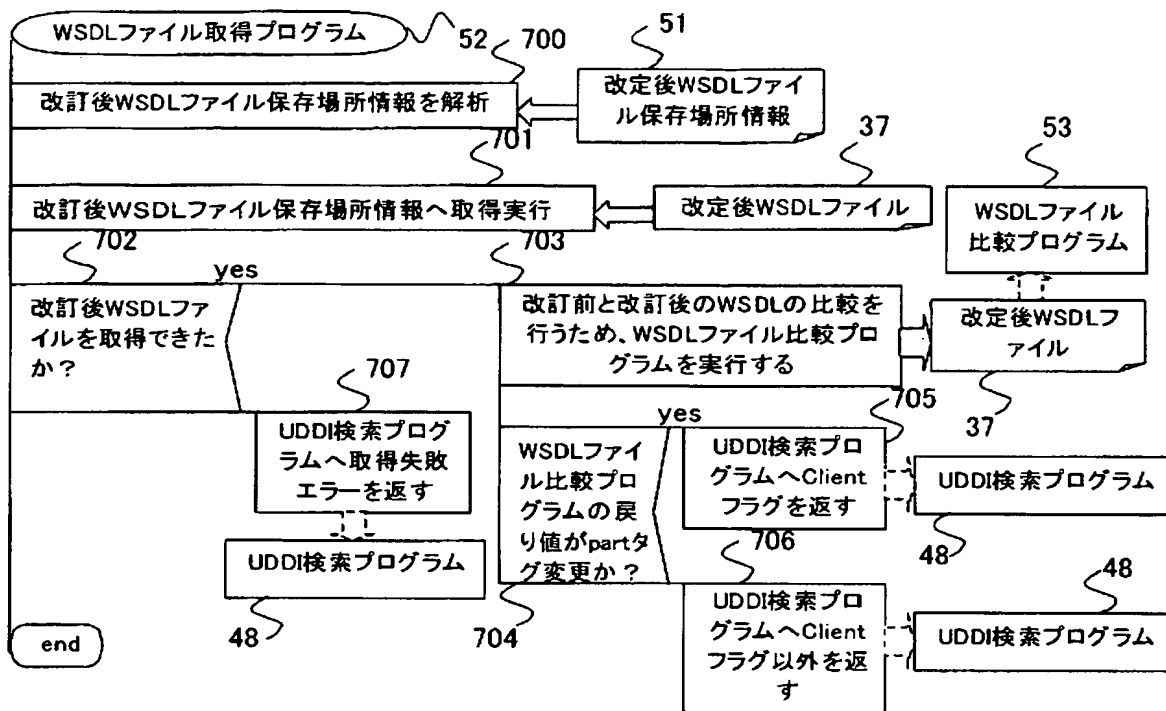


【図 2 2】



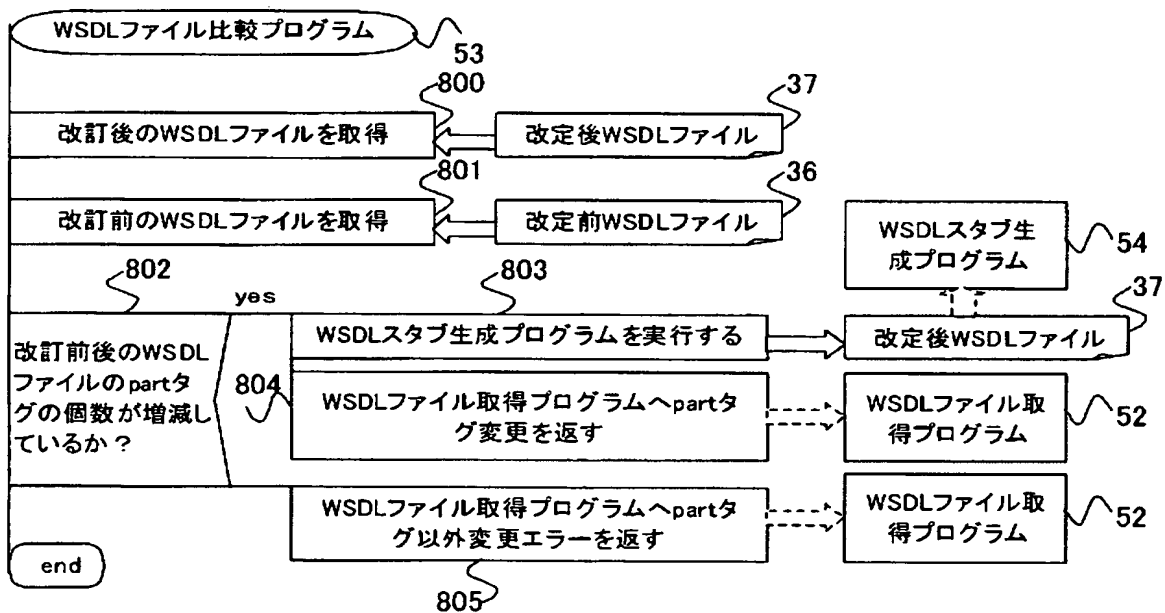
【図 23】

図 2 3



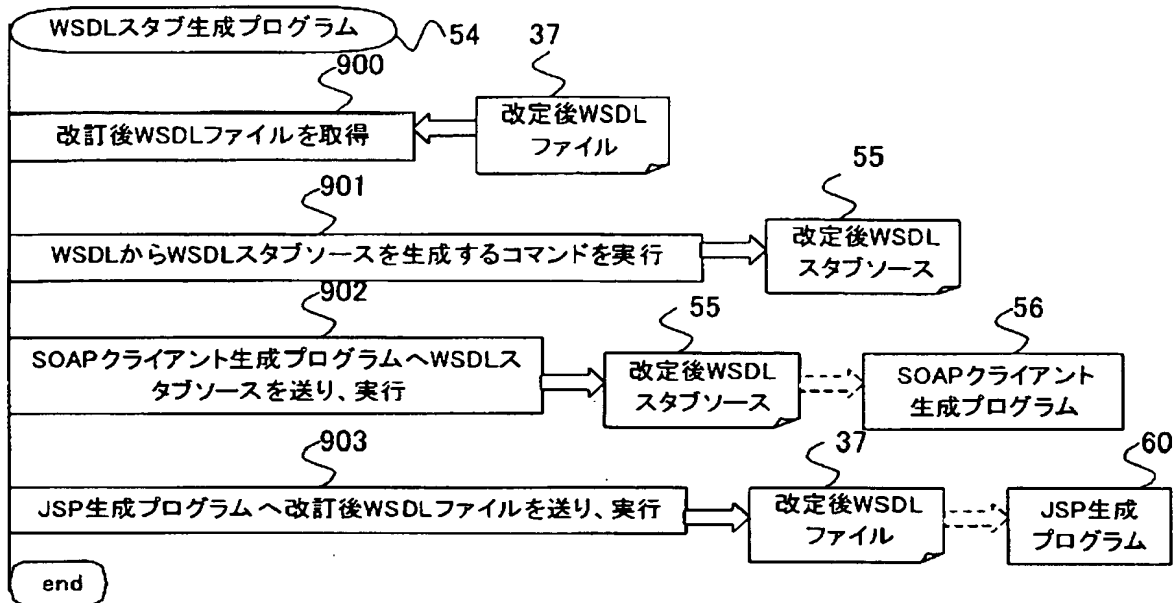
【図 2 4】

图 2 4



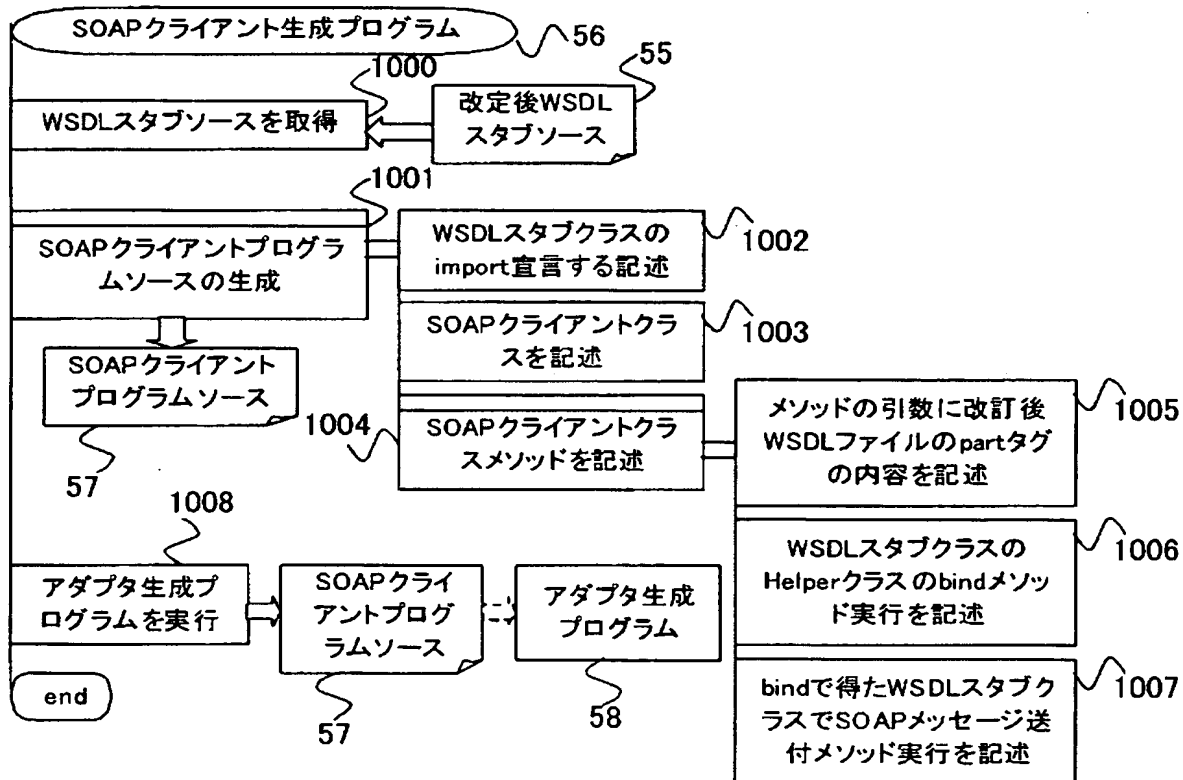
【図 2 5】

図 2 5

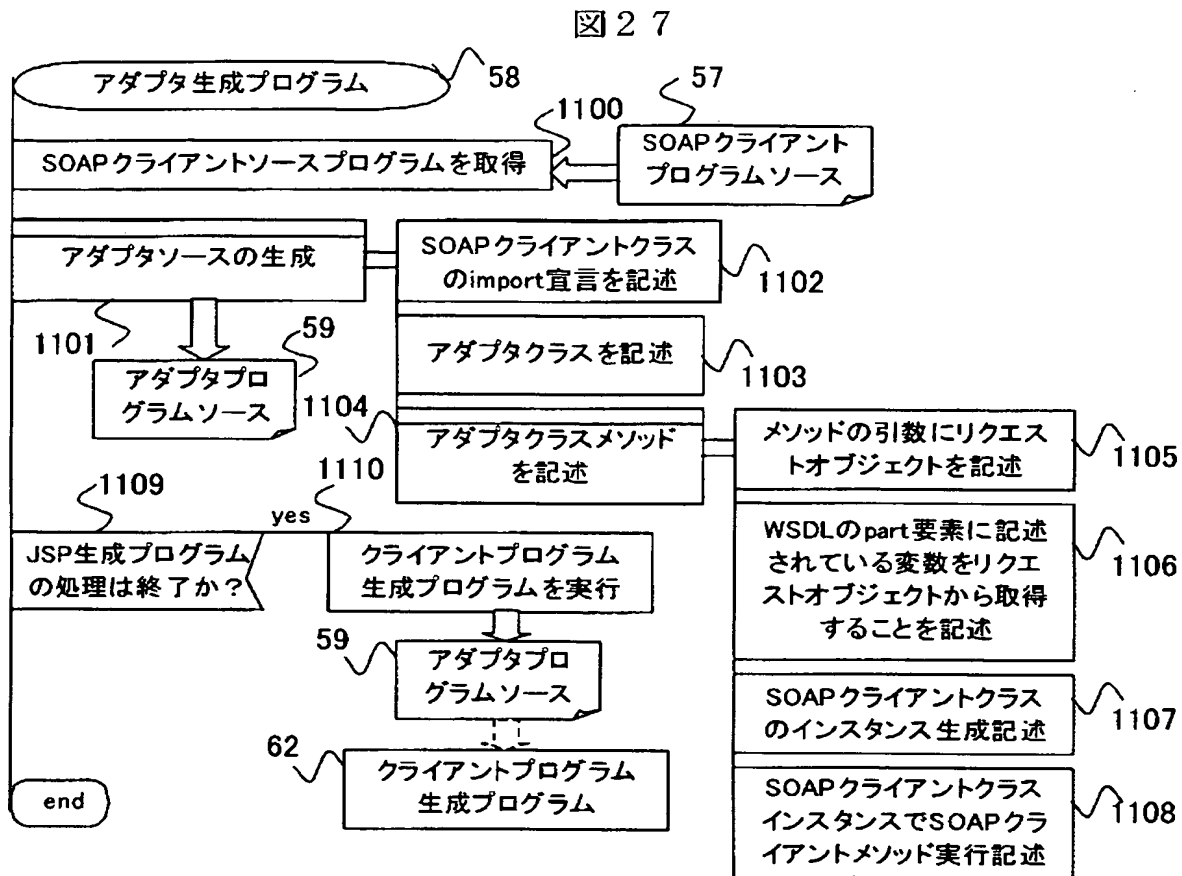


【図 2 6】

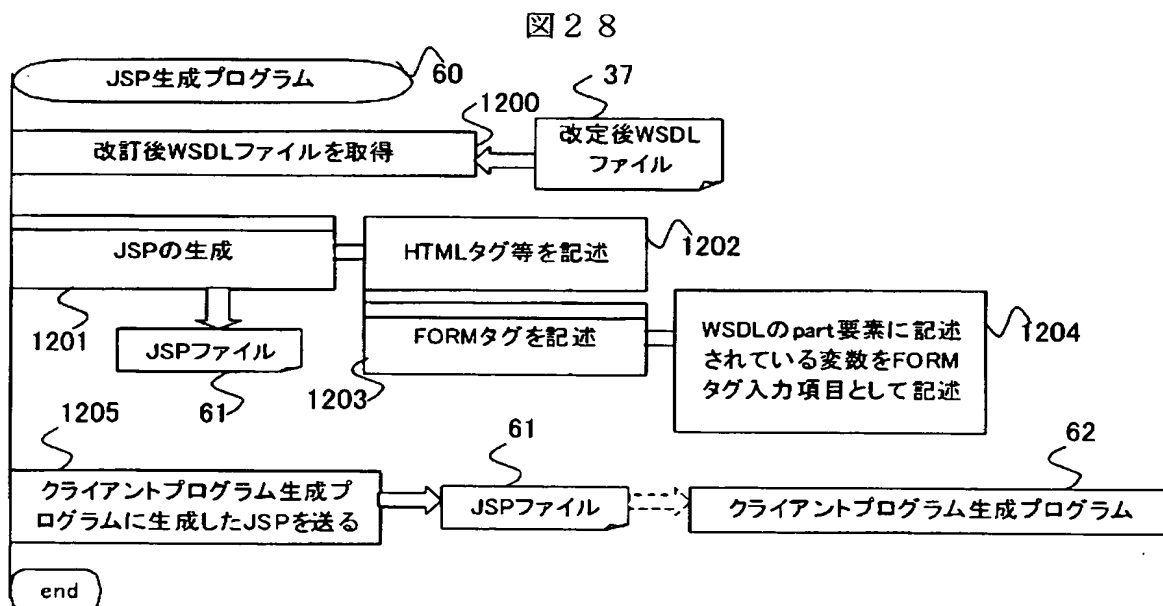
図 2 6



【図 27】



【図 28】



【書類名】 要約書

【要約】

【課題】

実行中のクライアントプログラムについて、インタフェースの変更による修正、再配置、停止、再実行等の変更管理を動的に行うことを可能とする。

【解決手段】

クライアントプログラムが、サーバプログラムを提供している所定URLへアクセス要求を送付して、エラーが発生したとき、このエラーの発生原因を解析するためのプログラムを起動し、インタフェース定義情報の変更によるエラーか否かを判定し、インタフェース定義情報の変更によるエラーの場合は、上記クライアントプログラムの変更箇所を解析するプログラムを起動し、変更箇所の特定制を行い、所定手続きに基づいて上記クライアントプログラムの上記変更箇所を変更して、上記クライアントプログラムを実行する。

【選択図】 図 1

認定・付加情報

特許出願の番号	特願 2 0 0 3 - 0 5 7 9 3 6
受付番号	5 0 3 0 0 3 5 3 2 1 9
書類名	特許願
担当官	小野寺 光子 1 7 2 1
作成日	平成 1 5 年 3 月 7 日

< 認定情報・付加情報 >

【提出日】	平成 15 年 3 月 5 日
-------	-----------------

次頁無

特願 2 0 0 3 - 0 5 7 9 3 6

出 願 人 履 歴 情 報

識別番号

[0 0 0 0 0 5 1 0 8]

1. 変更年月日

1 9 9 0 年 8 月 3 1 日

[変更理由]

新規登録

住 所

東京都千代田区神田駿河台 4 丁目 6 番地

氏 名

株式会社日立製作所